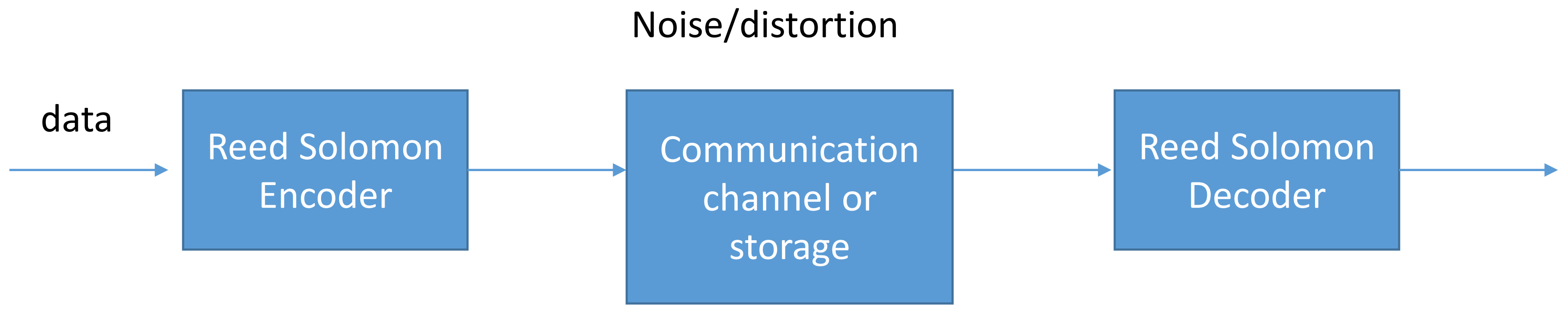
Reed Solomon Codes

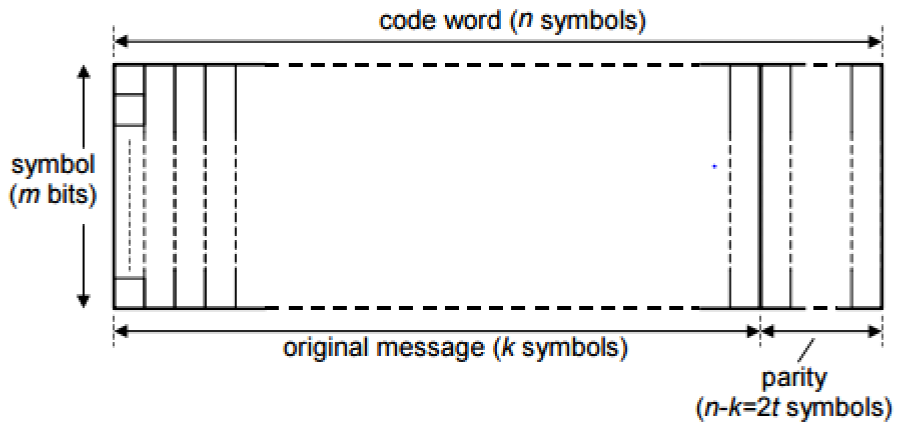
Jieyang Hu

Workflow


Noise/distortion
data
Reed Solomon Encoder
Communication channel or storage
Reed Solomon Decoder

Structure

- Linear block code
- Parameters n,k,q: n is the block symbol length, k is the message symbol length, and q is the size of each symbol in bits
- Each message and code symbol in our block corresponds to an element of a Galois field



Galois fields

- Denoted as GF(n), where n is the number of elements in the field
- A set of numbers with arithmetic operations add, subtract, multiply, and divide which are associatative, commutative, and closed
- Set of integers mod p where p is a prime form a prime field GF(p)
- Set of integers mod p^q form an extension field GF(p^q)
- For purposes of Reed Solomon codes, we are interested in GF(2^q).

Galois field

- Each element in a field can be expressed as a polynomial
- For GF(2^4) the polynomial is: $a_3x^3 + a_2x^2 + a_1x^1 + a_0x^0$
- Each coefficient is an element of GF(2), and is either 0 or 1
- The vector of coefficients ($a_{3,}a_{2,}a_{1,}a_0$) can be expressed as binary
- 1010 corresponds to the polynomial field element $x^3 + x$:
  $(1)x^3 + (0)x^2 + (1)x^1 + (0)x^0$

Addition and subtraction defined

- We add two field elements a and b by adding their polynomials:
  $(a_3x^3 + a_2x^2 + a_1x^1 + a_0x^0) + (b_3x^3 + b_2x^2 + b_1x^1 + b_0x^0) =$
  $(c_3x^3 + c_2x^2 + c_1x^1 + c_0x^0)$
  where each $c_i = a_i + b_i \bmod 2$
- Since each coefficient is either 0 or 1, for a given coefficient c_i, it will be 1 only if a_i and b_i differ. We can express addition as an XOR operation
- For example: adding $x^3 + x$ to $x^3 + x^2 + 1$
  is 1010 XOR 1101 gives 0111, or $x^2 + x + 1$
- Subtraction is identical to addition

Multiplication and division

- Can not simply multiply two polynomials, because result is no longer in the field
- For a given GF(2^q), need to find an irreducible polynomial p(x) of degree q. a(x) * b(x) is done mod p(x): multiply a and b, take the remainder
- For AES encryption in GF(2^8), that polynomial is $x^8 + x^4 + x^3 + x + 1$
- Note there can be many choices for a given field, but it must be agreed upon ahead of time
- Division is multiplication by the inverse, where inverse $a^{-1}(x)$ satisfies: $a(x) * a^{-1}(x) = 1$

Encoding

- For a Reed Solomon code with parameters n (block size), k (message size), q (symbol size in bits), we encode the message as a polynomial p(x), and then multiply with a code generator polynomial g(x)
- We construct code generator polynomial g(x) with n – k factors, each root being a consecutive element in the Galois field

$$g(x) = (x-\alpha)(x-\alpha^2)\cdots(x-\alpha^{n-k}) = g_0 + g_1 x + \cdots + g_{n-k-1}x^{n-k-1} + x^{n-k}\,.$$

- α is a primitive element, an alternative way of specifying elements in a field as successive powers 0, α^0 , α^1 , α^2 … α^N where $N = 2^q - 1$

Encoding

- For a Reed Solomon code with parameters n (block size), k (message size), q (symbol size in bits), we encode the message as a polynomial p(x), and then multiply with a code generator polynomial g(x)
- Map the message vector [x_1 … x_k] to a polynomial p(x) of degree < k such that:

  $p_x(a_i) = x_i$ holds for all $i = 1, \ldots, k$.
- Can be done using Lagrange interpolation. Once polynomial is found, evaluate at other for other points α_{k+1} … α_n
- Nice property that evaluating p at the first k points yields the original message symbols

Decoding

- Sender calculates s(x) = p(x) * g(x), and sends over the coefficients of s(x)
- Receiver receives r(x).
- Note that if s(x) == r(x), then r(x) / g(x) has no remainder
- Otherwise r(x) = p(x) * g(x) + e(x), where e(x) is an error polynomial
- Receiver knows g(x) and its roots. Can evaluate r(x) at every root to obtain a system of equations to determine which coefficients of r are in error, and the value of the error
- Berlekamp and Massey algorithm can solve
- Performs well for burst errors, since if a symbol errors in all 8 bits, the decoder will fix the entire symbol

Properties

- The message polynomial has degree $< k$, evaluated at n distinct points
- Two polynomials will agree in at most $k – 1$ points, so codeword will disagree in at least $n – (k – 1) = n – k + 1$ positions
- For two polynomials that do agree at $k – 1$ points, they will disagree at exactly $n – k + 1$ positions, so $n – k + 1$ is the minimum distance between code words
- A Reed Solomon decoder can correct up to t errors, where $2t = n – k$
- Intuitively, there are $n - k$ parity symbols, which are twice the amount of correctable errors. For each error, one redundant symbol is used to locate the error, and another redundant symbol is used to find its correct value.

Applications

- One of first implementations of Reed-Solomon was encoding pictures sent back from Voyager.
- Digital communication protocols such as DSL
- Data storage media such as DVDs and CDs
- Barcode formats such as QR codes

References

- http://downloads.bbc.co.uk/rd/pubs/whp/whp-pdf-files/WHP031.pdf BBC R&D white paper
- http://www.cs.cmu.edu/~guyb/realworld/reedsolomon/reed_solomon_codes.html
- https://math.berkeley.edu/~mhaiman/math55/reed-solomon.pdf
- https://en.wikipedia.org/wiki/Reed%E2%80%93Solomon_error_correction
- https://people.cs.clemson.edu/~westall/851/rs-code.pdf
- https://en.wikiversity.org/wiki/Reed%E2%80%93Solomon_codes_for_coders#Introduction_to_mathematical_fields
- https://www.youtube.com/watch?v=x1v2tX4_dkQ Introduction to Galois fields by Christof Paar